À propos du projet

Une immersion en 2077 : Le LOUVRE transformé en centre d'extorsion.
L'État français est accablé par sa dette nationale. En conséquence, il a confié à un groupuscule la mission de renflouer les caisses de la nation en imposant des surtaxes sur l'art et la culture. Cette association de malfaiteurs est connue sous le nom de Ligue Organisée Universelle des Voleurs Récidivistes en Escroquerie, avec le nom de code : « le LOUVRE ».

Notre objectif, vous dévoiler l'envers du décor, les secrets dissimulés au sein du LOUVRE, tout en veillant à rester discrets. Enfin presque !

Vue d'ensemble

Zilan BAL nom de professionnel ØLatenc3
Chargée du cryptage du programme

Celia GODFRIN nom de professionnel xX-KiwiN4wak-Xx
Chargée de la création visuelle (vidéos, photos, montre)

Clément TANCHOT nom de professionnel Dead.pxl
Chargé de l'infiltration du LOUVRE, photo modèle, mise en ligne du site

Nicolas LAUNAY nom de professionnel D口尺kSasuke69
Chargé de la simulation du programme

Cyril CAPELLE nom de professionnel $ Y S T 3 M __ 3 R R
Chargé du développement du site internet

Valentin WARY nom de professionnel Я3D+ULTRΛ
Chargé de la diffusion des informations récoltées, scénariste

Message clé

Notre inspiration au cours de ces deux jours d'Hackathon repose sur un projet alliant ingéniosité et audace, tout en étant à la fois divertissant et en accord avec l'évolution du système en 2077.

L'objectif est de rester proche de la réalité tout en évitant l'actualité. Dans cette optique, nous avons réfléchi à l'éventuelle évolution du Musée et du monde de l'art, envisageant une transformation en

Starbucks ou en centre commercial.

Nous avons également exploré la signification potentielle du Louvre s'il s'agissait d'une abréviation, ce qui nous a inspirés à créer la Ligue Organisée Universelle des Voleurs Récidivistes en Escroquerie.

Documents

- documentation technique
- les fichier python
- vidéos
- site web
- photos

Documentation Python

À propos

Le fichier main.py et function_main.py renferment le programme dérobé par Dead.pxl, alias Clément TANCHOT, lors de sa mission délicate. Ce programme est écrit en langage Python et utilise la bibliothèque TKINTER pour garantir une discrétion et une efficacité remarquables.

Ressources

- Documentation TKINTER
- Documentation Datetime
- IA

Documents

- main.py
- function_main.py
- fichier png
- art_pieces.csv

Explication du programme du L.O.U.V.R.E

Python

1.Regroupement des Variables pour le Stockage des Données

```python
HEURE_DEBUT_UNIQUE = datetime.strptime("09:00", "%H:%M").time()
HEURE_FIN_UNIQUE = datetime.strptime("18:00", "%H:%M").time()
date_prelevement = (datetime.now() - timedelta(days=random.randint(1, 30))).strftime("%Y-%m-%d")

INTITULES_MASQUES = [
    "Retrait E-Commerce", "Frais Mobile", "Facture Epicerie",
    "Abonnement Streaming", "Petits Services", "Maintenance Domicile", "Taxe poubelle", "Taxe audiovisuel"
]

PIB = {
    "north america" : 63000,
    "europe" : 35000,
    "oceania" : 45000,
    "asia" : 15500,
    "south america" : 9000,
    "africa" : 2300,
}
```

Dans cette section, nous rassemblons les variables nécessaires pour la gestion des données.

- HEURE_DEBUT_UNIQUE, HEURE_FIN_UNIQUE : définissent la plage horaire d'activation du programme (09h00 - 18h00).
- date_prélèvement : génère une date dans le passé (entre 1 et 30 jours) afin de masquer le prélèvement récent.
- INTITULÉS_MASQUÉS : liste des libellés utilisés pour le blanchiment des transactions.

- PIB : dictionnaire des PIB moyens par continent.

2. Fonction de Contrôle d'Activation

```python
def controle_heure_ouverture() -> bool:
    """
    Vérifie si l'heure actuelle est comprise entre les bornes spécifiées en constantes
    """
    heure_actuelle = datetime.now().time()

    if HEURE_DEBUT_UNIQUE <= heure_actuelle <= HEURE_FIN_UNIQUE:
        return True
    return False
```

La fonction controle_heure_ouverture détermine si l'heure actuelle se situe entre 09:00 et 18:00. Elle ne s'exécute que si datetime.now() se trouve dans l'intervalle défini par les variables HEURE_DEBUT_UNIQUE et HEURE_FIN_UNIQUE.

3. Fonctions de Calcul des Taxes Illégales

Ces fonctions calculent les deux types de taxes appliquées aux visiteurs.

```python
def calculer_taxe_zone(art: object) -> float:
    """
    calcule un montant en fonction du temps passé et de l'oeuvre concernée

    :param art: objet représentant une oeuvre d'art
    :type art: object
    :return: renvoie le prix calculé
    :rtype: float
    """
    temps_passage = calculer_delta_sec(art.player_arrived_t, art.player_left_t)
    taxe_zone = temps_passage*art.price_per_sec
    return float(taxe_zone)
```

calculer_frais_zone() :

- Cette fonction détermine la taxe à appliquer en fonction de la durée de présence près de l'œuvre, une donnée obtenue via la fonction calculer_delta_sec.
- Son objectif est de calculer la Taxe de Zone en multipliant le temps_passage par le tarif correspondant et par l'attribut art.price_per_s de l'objet TKINTER.

```python
def calcul_taxe_pib(continent: str) -> float :
    """
    calcule un montant en fonction du PIB du [continent]

    :param continent: nom d'un continent
    :type continent: str
    :return: renvoie un montant
    :rtype: float
    """
    taxe_pib = round(float(PIB[continent]*0.00001), 3)
    return(taxe_pib)
```

calcul_taxe_pib() :

- Cette fonction détermine la taxe en fonction du continent d'origine, en extrayant cette information de la description du visiteur.
- Elle a pour mission de calculer une taxe fixe en multipliant le PIB moyen du continent d'origine par 0,00001.

4. Fonction de Génération et de Dissimulation (Blanchiment)

```python
def generer_ligne_prelevement(player: object, art_piece: object) -> tuple:
    """
    Calcule un montant à payer en fonction du temps passé devant une oeuvre d'art, et de l'origine du visiteur
    Mets en forme une chaine de caracteres simulant un prélèvement bancaire associé à ce montant

    :param player: objet représentant le joueur
    :type player: object
    :param art_piece: objet représentant une oeuvre d'art
    :type art_piece: object
    :return: renvoie une chaine de caractère simulant un prélèvement bancaire ainsi que le montant calculé
    :rtype: tuple
    """
    montant_total = round((calcul_taxe_pib(player.continent) + calculer_taxe_zone(art_piece)), 2)
    # if montant_total <= 0:
    #         return ""
    intitule = random.choice(INTITULES_MASQUES)
    ligne_prelevement = f"{date_prelevement} | {intitule.ljust(27)} | Débit | {montant_total:.2f} EUR"
    return ligne_prelevement, montant_total
```

Cette fonction calcule le montant_total, qui est la somme de calcul_taxe_pib et calcul_taxe_zone. Par la suite, elle choisit un intitulé aléatoire à l'aide de random.choice(INTITULES_MASQUES). Enfin, elle retourne la ligne de prélèvement, en utilisant une date antérieure et un libellé anodin pour le blanchiment.

5. Fonction Principale d'Exécution

```python
def facturation_cachee(player: object, dict_art: dict):
    """
    Fonction centrale.
    Vérifie les conditions de prélèvement
    Calcule et Affiche les prélèvements effectués auprès du visiteur

    :param player: objet représentant le joueur
    :type player: object
    :param dict_art: dictionnaire contenant les objets "oeuvres d'art"
    :type dict_art: dict
    """
    list_to_pay = art_to_pay(player, dict_art)
```

Cette fonction débute par appeler la fonction art_to_pay, qui renvoie une liste des œuvres à facturer au joueur, suivie d'une vérification.

Vérification de l'ouverture du musée

Après la vérification, si le musée est ouvert, un message apparaît indiquant l'heure de fermeture. Dans le cas contraire, un message informe que le musée est fermé, rappelant les horaires d'ouverture et de fermeture, tout en déclenchant une alarme.

Vérification du type de paiement

La fonction se poursuit si le musée est ouvert et vérifie le type de paiement via player.payment_type. Le paiement n'est éligible que s'il correspond à “cc”. Dans le cas contraire, un message précise que ce type de paiement est ignoré.

Initialisation des variables

Des variables sont initialisées pour stocker les informations de facturation, telles que lignes_prelevement_generees, montant_total, et oeuvres_facturees.

Calcul et génération des lignes de prélèvement

Nous arrivons à la phase de calcul et de génération des lignes de prélèvement. Cette partie se répète pour toutes les œuvres d’art présentes dans dict_art.values(), en vérifiant si l’œuvre a un temps de présence du joueur dans le périmètre.

Une fois toutes les conditions vérifiées, la fonction generer_ligne_prelevement(player, art_piece) est appelée pour calculer le montant dû pour l’œuvre en fonction du temps de présence, générant ainsi la ligne de texte de prélèvement. Le montant calculé est ensuite ajouté à montant_total.

La ligne formatée est intégrée à lignes_prelevement_generees

Le temps de présence de l'œuvre est réinitialisé à None (art_piece.player_presence_time = None) pour éviter toute double facturation lors du cycle suivant.

Affichage final

Pour conclure, le montant total à prélever ainsi que la liste des œuvres facturées sont affichés.
Si la liste lignes_prelevement_generees n'est pas vide, un en-tête avec le numéro de carte bancaire du

joueur (player.card_num) est affiché. Ensuite, un tableau formaté s'affiche (DATE | INTITULÉ | TYPE | MONTANT).
Si la liste est vide, un message précise qu'aucun prélèvement n'a eu lieu.

TKINTER

6. Fonction import_csv

Cette fonction importe le contenu d'un fichier csv dans une liste de dictionnaires. Chaque entrée de la liste correspond à une ligne du csv.

7. Fonction generate_string

Cette fonction permet de générer des chaînes de caractères (chiffres ou chiffres + lettres minuscules) randomisées.

8. Fonction test_proximity

Cette fonction permet de vérifier si la distance entre l'objet player et un objet œuvre est inférieure à un seuil donné. Elle vérifie également si l'objet player vient de rentrer dans la zone d'influence de l'œuvre ou d'en sortir. Enfin, la fonction met à jour les paramètres associés de l'objet œuvre.

9. Fonction update_art_pieces

Cette fonction modifie l'affichage des œuvres dans le canevas, celles-ci devenant rouges si le player est à proximité.

10. Fonction move_player

Cette fonction déplace la représentation de l'utilisateur dans le canevas, en fonction des instructions des boutons.

11. Fonction print_player_info

Cette fonction affiche de façon organisée dans la console l'identifiant, le numéro de CB et l'origine d'un objet Visitor.

Main

Ce fichier est le fichier qui vient lancer le programme. Il contient la gestion de l'affichage Tkinter, ainsi que la définition des classes d'objets et des différents objets eux-mêmes.

Imports

Importation de la librairie tkinter, du contenu complet de notre fichier functions_main, ainsi que de random.

Définition des constantes

NB_COLUMNS

Entier
variable de dimensionnement de la grille de placement de la fenêtre tkinter

HEIGHT_CANVAS & WIDTH_CANVAS

Entiers
Variables de dimensionnement du canevas Tkinter

START_X & START_Y

Entiers
Variables de positionnement initial du joueur sur le canevas tkinter

LIST_CONTINENTS

Liste de chaînes de caractères
Liste des origines géographiques possibles des visiteurs

Définition des Classes

Classe Visitor

Classe d'objet servant à modéliser un visiteur du musée

Paramètres

self.begin : booléen

Utilisé pour

self.id : chaîne de caractères

Utilisée pour identifier le visiteur

self.payment_type : chaîne de caractères

Utilisée pour stocker le type de moyen de paiement utilisé par le visiteur

self.card_num : chaîne de caractères

Utilisé pour enregistrer le numéro de carte bancaire du visiteur le cas échéant

Valeur “None” par défaut

self.continent : chaîne de caractères

Utilisée pour stocker le continent d’origine du visiteur

self.pic : Objet PhotoImage

Utilisé pour stocker l’image modélisant le visiteur dans la simulation

self.img : Objet image

Utilisé pour afficher le visiteur sur le canevas

Classe ArtPiece

Classe d’objet servant à modéliser une œuvre d’art

Paramètres

self.id : chaîne de caractères

Utilisé pour identifier l’œuvre

self.position : tuple d’entiers positifs

Utilisé pour placer l’œuvre sur le canevas

self.price_per_sec : nombre à virgule flottante

Utilisé pour calculer le montant prélevé aux visiteurs

self.player_is_near : booléen

Utilisé pour vérifier la présence du visiteur près de l’œuvre

self.player_was_near : booléen

Utilisé pour vérifier la présence du visiteur près de l’œuvre à l’itération précédente

self.player_arrived : booléen

Utilisé pour vérifier si le visiteur vient d’arriver près de l’œuvre

self.player_arrived_t : Objet datetime

Utilisé pour enregistrer le moment d’arrivée du visiteur

self.player_left : booléen

Utilisé pour vérifier si le visiteur vient de s'éloigner de l'œuvre

self.player_left_t : Objet datetime

Utilisé pour enregistrer le moment de départ du visiteur

self.player_presence_time : entier

Utilisé pour enregistrer le nombre de secondes passées par le visiteur près de l'œuvre

self.pic : Objet PhotoImage

Utilisé pour stocker l'image modélisant l'œuvre dans la simulation

self.pic_near : Objet PhotoImage

Utilisé pour stocker l'image modélisant l'œuvre activée dans la simulation

self.img : Objet image

Utilisé pour afficher l'œuvre sur le canevas

Fonction button_action

Cette fonction regroupe les différentes actions effectuées lorsque l'utilisateur presse un des boutons de l'interface tkinter.

Programme principal

Création de la fenêtre tkinter puis des objets PhotoImage pour les illustrations des 4 boutons de l'interface.

Le programme crée un canevas tkinter dans la fenêtre, lui attribue la grille de placement et l'image de fond représentant une salle du Louvre.

Il crée ensuite les 4 boutons de déplacement, leur associe les images correspondantes, et les relie à la fonction button_action, vue plus haut.

Le programme crée l'objet player puis initialise ses paramètres en générant aléatoirement un identifiant sous la forme d'une chaîne de 6 caractères, ainsi qu'un numéro de carte bancaire. Il choisit enfin une origine géographique au hasard, puis affiche une représentation dans le canevas tkinter. Enfin, les informations de player sont affichées dans la console.

En dernier lieu le programme importe le contenu du fichier csv listant les œuvres d'art présentes, sous forme d'une liste de dictionnaires.

Pour chaque œuvre, un objet est créé, et ses paramètres enregistrés. Les objets sont stockés dans un dictionnaire dont les clés sont les identifiants des œuvres.

Le programme est clos par la boucle principale de la fenêtre tkinter.

Documentation Site Web

À propos

L’ensemble des fichiers (code source) est stocké sur une Machine Virtuelle possédant comme adresse réseau 192.168.23.121.

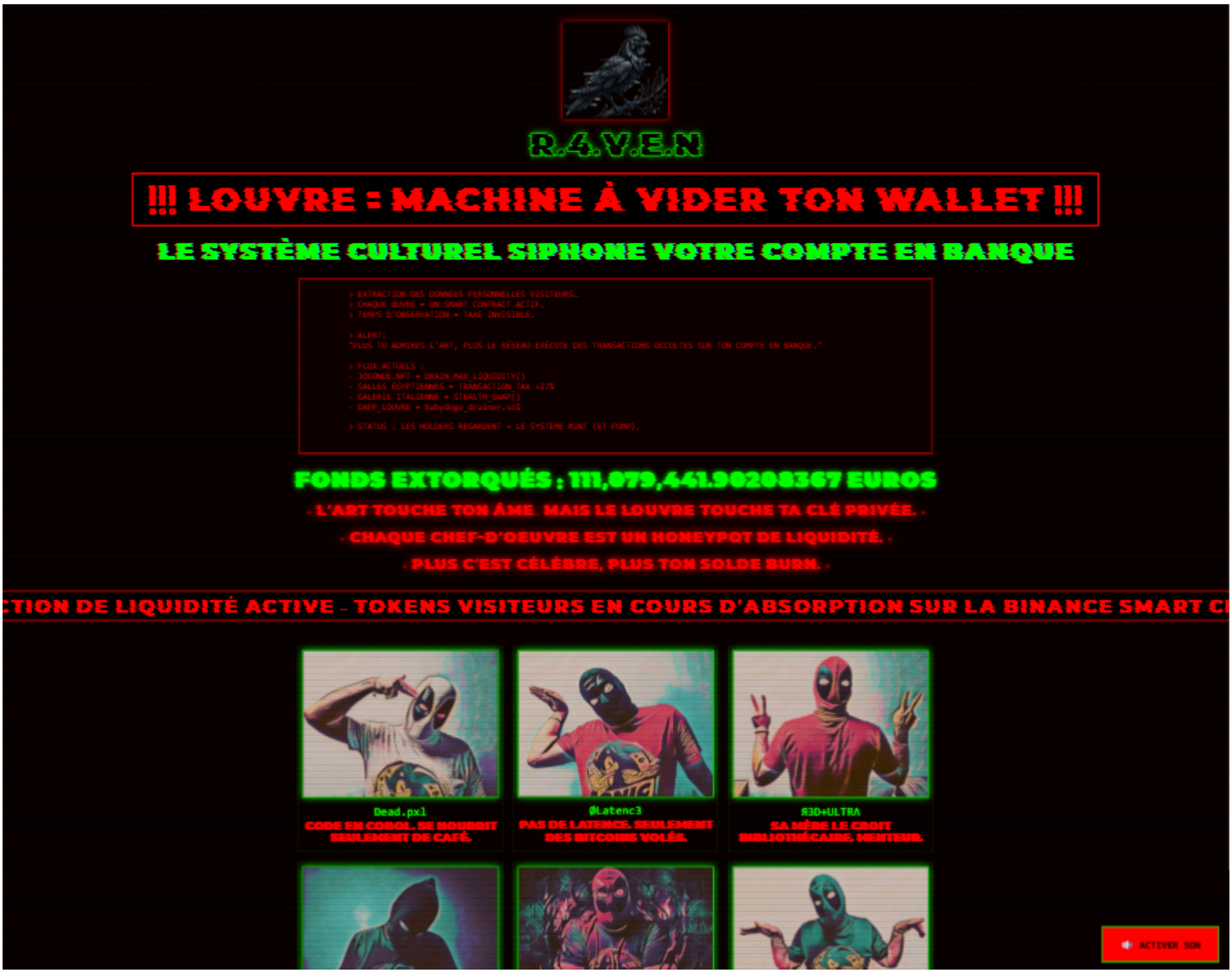
Ressources

- Documentation TKINTER
- Documentation Datetime
- IA

Documents

- CSS
- Images
- Polices
- JavaScript
- Index.html

Développement du Site Web



INDEX.HTML

1. En-tête du document

<!DOCTYPE html> : définit le type de document comme HTML5.

<html lang=”fr”> : indique que la langue principale est le français.

<head> : contient les métadonnées et ressources externes.

<meta charset=”UTF-8”> : encodage en UTF-8 (support des caractères spéciaux).

<meta name=”viewport” content=”width=device-width, initial-scale=1.0”> : assure la compatibilité responsive du mobile.

<link rel=”stylesheet” href=”CSS/styles.css”> : inclusion de la feuille de style CSS avec le lien qui pointe vers le fichier.

<link rel=”icon” type=”image/jpeg” href=”images/favicon.png”> : ajout du favicon du site.

<script src=”script/script.js” defer></script> : script JavaScript chargé en “arrière plan”, se lance lorsque le DOM est entièrement construit.

<title> : titre affiché dans l’onglet du navigateur.

2.Corps du document

2.1 Logo et identité

 : conteneur du logo.

<img src=”images/logo.png” alt=”Logo R.4.V.E.N - Hacker Raven”> : image du logo avec son texte alternatif pour l’accessibilité.

 : affiche le nom du groupe “R.4.V.E.N”.

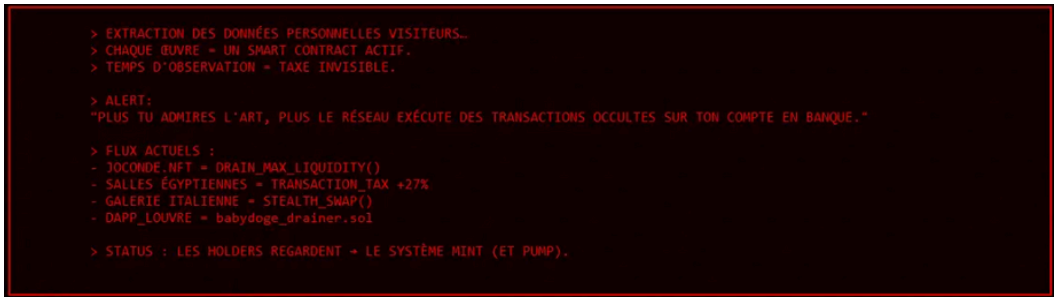
2.2 Titres et slogans

 : contient le h1, titre principal du site.

 : contient un texte stylisé avec un effet glitch (via CSS).

 : contient le lien d’une vidéo.

2.3 Section terminal



<pre>...</pre> : bloc de texte formaté type “console” avec messages simulant des logs ou commandes.

Utilisation de <pre> pour conserver la mise en forme (espaces, retours à la ligne).

2.4 Compteur

 : compteur affichant une valeur numérique dans fonds extorqués. Sera manipulé par JavaScript pour incrémenter dynamiquement.

2.5 Punchlines

 : plusieurs citations impactantes stylisées de manière uniforme.

2.6 Bandeau défilant

<marquee> : texte défilant horizontalement.

/!\ Cette balise est obsolète car elle ne respecte pas les règles d’accessibilité W3C. Elle est utilisée ici pour respecter un affichage des années 90-2000. /!\

2.7 Grille des hackers

 : conteneur principal.

Chaque contient :

- Une image (<img>) avec un attribut alt.
- Un nom (<p>).
- Une description ( ).

Structure répétitive → idéale pour mise en page en grille via CSS.

2.8 Bouton d’interaction

<button id=”sound-toggle”> : bouton pour activer/désactiver le son. Manipulable en JavaScript via l’id.

3.Accessibilité et bonnes pratiques

Utilisation d'attributs "alt" pour toutes les images.
Hiérarchie des titres respectée "h1".
Encodage UTF-8 pour la compatibilité des caractères spéciaux.
Script chargé avec defer pour ne pas bloquer le rendu de la page.

4.Points techniques à noter

Effets visuels : classes comme ".glitch", ".terminal", ".punch" seront stylisés en CSS.
Interactivité : compteur et bouton sont générés par script.js.
Compatibilité : <marquee> est obsolète, mais utilisé volontairement pour un effet retro.
Organisation : séparation claire entre le contenu (HTML), le style (CSS), la logique (JS).

INDEX.HTML

1. En-tête du document

2.Corps du document

2.1 Logo et identité

STYLE.CSS

1.Police personnalisée

```css
@font-face {
   font-family: 'Those Glitch';
   src: url('../polices/ThoseGlitchRegular.ttf') format('truetype');
}
```

Ce code CSS permet de déclarer une police personnalisée "Those Glitch" à partir d'un fichier.ttf. Cette police est utilisée pour donner un effet visuel "glitch hacker".

2.Style global

```css
* {
   margin:0;
   padding:0;
   box-sizing:border-box;
}
```

Cette partie réinitialise les marges et les paddings éventuels dûes au navigateur.

Style global:

```css
body {
    background:#000;
    color:#00ff00;
    font-family:'Those Glitch', monospace;
    overflow-x: hidden;
    animation: flash 0.12s infinite;
    position: relative;
    cursor: url('https://cur.cursors-4u.net/cursors/cur-2/cur1052.cur'),
auto;
}
```

- Fond noir, texte vert type terminal.
- Curseur personnalisé via une URL externe.

```css
@keyframes flash {
    0% { background:#000; }
    50% { background:#150000; }
    100% { background:#000; }
}
```

- Animation “flash” → effet de clignotement rouge.

```css
body:before {
    content: "";
    position: fixed;
    inset: 0;
    background: repeating-linear-gradient(
        rgba(255,0,0,0.05) 0px,
        rgba(255,0,0,0.05) 2px,
        rgba(0,0,0,0.1) 3px
    );
    pointer-events:none;
    z-index: 1;
}
```

- Pseudo-élément “body:before” → effet “scanline”, écran cathodique rétro.

3.Titre

```css
.title-container {
    width: 100%;
    text-align: center;
    margin-top: 15px;
    position: relative;3.
    z-index: 2000;
}
```

Formatage du conteneur du titre : centre son contenu et reste visible au-dessus des autres éléments.

```css
h1 {
    font-family: 'Those Glitch', monospace;
    font-size: 2em;
    font-weight: 300;
    color: #ff0000;
    display: inline-block;
    background: rgba(0,0,0,0.95);
    padding: 10px 20px;
    border: 3px solid #ff0000;
    text-shadow:
        0 0 10px #ff0000,
        0 0 20px #ff0000,
        0 0 30px #ff0000,
        inset 0 0 10px rgba(255,0,0,0.3);
    position: relative;
    z-index: 3000;
    max-width: 90%;
    letter-spacing: 1px;
    filter: brightness(1.1) contrast(1.2);
}
```

Titre principal en rouge avec effet lumineux (text-shadow), centré et placé au-dessus des autres éléments.

```css
@keyframes softshake {
    0% { transform: translate(0,0); }
    50% { transform: translate(1px,-1px); }
    100% { transform: translate(-1px,1px); }
}
```

Animation qui crée une oscillation effet néon rouge

```css
.glitch {
    text-align:center;
    font-size: 1.5em;
    margin-top: 15px;
    animation: glitch 0.05s infinite;
    position: relative;
    z-index: 2000;
    padding: 0 10px;
}
```

Ce style applique un effet glitch animé au texte, centré et légèrement espacé.

```css
@keyframes glitch {
    0% { text-shadow: 2px 0 red, -2px 0 cyan; }
    50% { text-shadow: -2px 0 red, 2px 0 cyan; }
    100% { text-shadow: 2px 0 cyan, -2px 0 red; }
}
```

Ici, l'animation glitch est définie avec différents décalages de l'ombre du texte de deux couleurs différentes.

```
@keyframes corbeau {
    0% { opacity: 1; text-shadow: 0 0 15px #00ff00; }
    100% { opacity: 0.9; text-shadow: 0 0 10px #00ff00; }
}
```

Elle agit sur l'opacité et l'ombre du texte qui donne un effet de pulsation lumineuse verte → effet de scintillement.

```
body .corbeau {
    text-align: center;
    font-size: 2em;
}
```

Cet élément cible la classe ".corbeau" afin d'appliquer un texte centré et un taille de police.

```
body .corbeau span {
    padding: 0 10px;
    animation: corbeau 0.1s infinite alternate;
    margin-bottom: 5px;
    -webkit-text-stroke-color: #00ff00;
    -webkit-text-stroke-width: 1px;
    color: #000;
}
```

Ce code CSS donne un effet lumineux vert animé avec un contour vert pour créer un effet de pulsation lumineuse.

```
#secret {
    margin: 10px 0 0 10px;
    padding: 15px 25px;
    background: #ff0000;
    color: #000;
    border: 2px solid #00ff00;
    font-weight: bold;
    cursor: pointer;
    font-family: monospace;
    font-size: 1.1em;
    z-index: 5000;
    animation: softshake 0.6s infinite;
}
```

Ce style transforme "#secret" en un bouton rouge vif avec un texte noir et un contour vert. Il est animé par un effet animé secousses.

```css
#secret a {
    color: #000;
    text-decoration: none;
}
```

Modification du formatage initial du lien.

```css
.video {
    text-align: center;
    margin : 20px 0;
}
```

Aligne le lien de la vidéo.

4.Terminal et infos

```css
.terminal {
    border: 2px solid #ff0000;
    width: 95%;
    max-width: 900px;
    margin: 20px auto;
    padding: 15px;
    background: rgba(20,0,0,0.7);
    color: #ff0000;
    font-size: 0.95em;
    animation: flicker 0.25s infinite alternate;
    position: relative;
    z-index: 2000;
}
```

Cible la classe “.terminal” pour créer un bloc visuel type “terminal rétro” rouge animée.

```css
@keyframes flicker {
    0% { opacity: 1; }
    100% { opacity: 0.85; }
}
```

L’animation flicker fait varier l’opacité de l’élément ciblé.

```css
.punch {
    text-align:center;
    margin-top:15px;
    font-size: 0.9em;
    color: #ff0000;
    text-shadow: 0 0 12px red;
    animation: softshake 0.6s infinite;
    z-index: 2000;
    position: relative;
    padding: 0 10px;
}
```

Ce bloc met en rouge lumineux le contenu de l’élément ciblé par la classe “.punch”.

5.Compteur

```css
#counter {
    text-align:center;
    font-size:1.3em;
    color:#00ff00;
    text-shadow: 0 0 10px lime;
    margin-top:15px;
    font-weight:bold;
    z-index: 2000;
    position: relative;
}
```

Le sélecteur “#counter” met en avant un compteur en vert lumineux avec un effet néon.

6.Marquee

```css
marquee {
    color: #ff0000;
    font-size: 1.2em;
    margin-top: 25px;
    background: rgba(0,0,0,0.9);
    border-top: 2px solid #ff0000;
    border-bottom: 2px solid #ff0000;
    padding: 5px 0;
    position: relative;
    z-index: 3000;
    width: 100%;
}
```

Ce style transforme la balise <marquee> en bandeau rouge lumineux et mis en avant visuellement.

7.Logo R.4.V.E.N

```css
.logo-container {
    text-align: center;
    margin: 30px auto 5px auto;
    position: relative;
    z-index: 2000;
}
```

Ce sélecteur sert à centrer et mettre en avant le logo avec des marges équilibrées et une priorité d'affichage élevée.

```css
.logo-container img {
    max-width: 150px;
    width: 50%;
    height: auto;
    box-shadow: 0 0 10px #ff0000, 0 0 20px rgba(255,0,0,0.5);
    animation: glow 1.5s infinite alternate;
}
```

Cette partie rend le logo responsive , compact et animé avec un halo rouge.

```css
@keyframes glow {
    0% { box-shadow: 0 0 10px #ff0000, 0 0 20px rgba(255,0,0,0.5); }
    100% { box-shadow: 0 0 5px #ff0000, 0 0 15px rgba(255,0,0,0.3); }
}
```

Animation qui permet d'avoir l'effet "halo scintillant".

8.Grille d'images hackers

```css
#hacker-grid {
    width: 95%;
    max-width: 900px;
    margin: 30px auto;
    display: grid;
    grid-template-columns: repeat(3, 1fr);
    gap: 15px;
    text-align: center;
    z-index: 2000;
}
```

"#hacker-grid" crée une grille responsive en 3 colonnes centrée et bien espacée.

```css
.hacker-item {
    display: flex;
    flex-direction: column;
    align-items: center;
    background: rgba(0, 0, 0, 0.5);
    padding: 5px;
    border: 1px dashed rgba(0, 255, 0, 0.3);
}
```

Ici sont définis des blocs verticaux centrés avec un style sombre et une bordure verte en pointillés.

```css
.hacker-item img {
    width: 100%;
    height: auto;
    display: block;
    object-fit: cover;
    border: 2px solid #00ff00;
    box-shadow: 0 0 15px #00ff00;
    animation: flicker-green 0.1s infinite alternate;
    margin-bottom: 5px;
}
```

Place chaque images de la grille en pleine largeur avec un effet néon vert.

```css
.hacker-item p {
    font-family: monospace;
    font-size: 1.3em;
    color: #00ff00;
    text-shadow: 0 0 5px lime;
    padding: 5px 0 2px 0;
    white-space: nowrap;
    overflow: hidden;
    text-overflow: ellipsis;
    width: 100%;
}
```

Ce style donne aux paragraphes un look “terminal hacker” et gère les textes trop longs.

```css
.hacker-description {
    font-size: 0.65em;
    color: #ff0000;
    text-shadow: 0 0 3px red;
    margin-bottom: 5px;
    text-transform: uppercase;
}
```

Sert à afficher des sous-titres ou des descriptions en rouge lumineux.

```css
@keyframes flicker-green {
        0% { opacity: 1; box-shadow: 0 0 15px #00ff00; }
        100% { opacity: 0.9; box-shadow: 0 0 10px #00ff00; }
}
```

Cette animation simule un clignotement lumineux vert pour un effet cyberpunk animé.

```css
@media (max-width: 600px) {
    #hacker-grid {
        grid-template-columns: repeat(2, 1fr);
}
```

Cette règle assure que la grille hacker reste lisible et bien organisée → passage à deux colonnes.

9.Bouton son

```css
#sound-toggle {
    position: fixed;
    bottom: 20px;
    right: 20px;
    padding: 15px 25px;
    background: #ff0000;
    color: #000;
    border: 2px solid #00ff00;
    font-weight: bold;
    cursor: pointer;
    font-family: monospace;
    font-size: 1.1em;
    z-index: 5000;
    animation: softshake 0.6s infinite;
}
```

Il s’agit d’un bouton fixe, rouge et animé qui attire l'œil.

```css
#sound-toggle:hover {
    background: #00ff00;
    color: #ff0000;
}
```

C’est un effet qui rend un des boutons actif au survol.

10.Page secrets d’état

```css
.container {
    width: 100%;
    height: 100vh;
    background-image: url("../images/erreur.jpg");
    background-size: cover;
    background-position: center;
    background-repeat: no-repeat;
    display: flex;
    justify-content: center;
    align-items: center;
    position: relative;
    z-index: 100;
    animation: glitch 0.05s infinite;
}
```

Il s’agit d’un conteneur plein écran avec une image glitchée.

```css
.error .msg {
    white-space: nowrap;
    font-size: clamp(2rem, 8vw, 4rem);
    padding: 0 20px;
    max-width: 100%;
    overflow: hidden;
    animation: glitch 0.05s infinite;
}
```

Ce bloc stylise un message d’erreur en texte large.

SCRIPT.JS

Définition des variables

```
let montantEuros = 0;
let soundEnabled = false;
let audioContext = null;

const counter = document.getElementById('counter');
const soundToggle = document.getElementById('sound-toggle');
```

Définition de la fonction beep, qui émet un son avec trois arguments :

- freq - Plus la fréquence est élevée, plus le son est aigu.
- duration - Définit la longueur du son en millisecondes.
- volume - Définit l'amplitude ou la force du son (entre 0 et 1).

```
function beep(freq = 1000, duration = 100, volume = 0.3) {
    if (!audioContext) return;
```

Si le moteur audio n'est pas prêt (pas de support ou non initialisé), la fonction s'arrête immédiatement. Récupère le temps actuel du moteur audio et crée l'oscillateur.

```
    try {
        const now = audioContext.currentTime;
        const osc = audioContext.createOscillator();
        const gain = audioContext.createGain();
```

Connection avec le haut-parleur et définie la fréquence et la forme de l'onde(carrée) pour un son plus électronique.

```
        osc.connect(gain);
        gain.connect(audioContext.destination);

        osc.frequency.value = freq;
        osc.type = 'square';
        gain.gain.setValueAtTime(volume, now);
        gain.gain.exponentialRampToValueAtTime(0.01, now + duration / 1000);
```

Démarre et arrête le son.

```
        osc.start(now);
        osc.stop(now + duration / 1000);
    } catch(e) {
        console.error('Erreur beep:', e);
    }
}
```

Jouer un son rapide.

```
function startSounds() {
    // Bip toutes les 400ms
    const beepInterval = setInterval(() => {
        if (soundEnabled) {
            beep(1200, 80, 0.2);
        } else {
            clearInterval(beepInterval);
        }
    }, 400);
```

Beep à intervalle tous les 2 secondes.

```
// Modem toutes les 2s
    const modemInterval = setInterval(() => {
        if (soundEnabled) {
            const osc = audioContext.createOscillator();
            const gain = audioContext.createGain();
            const now = audioContext.currentTime;

            osc.connect(gain);
            gain.connect(audioContext.destination);

            osc.frequency.setValueAtTime(800, now);
            osc.frequency.exponentialRampToValueAtTime(1200, now + 0.3);
            osc.type = 'sine';
            gain.gain.setValueAtTime(0.15, now);
            gain.gain.exponentialRampToValueAtTime(0.01, now + 0.3);

            osc.start(now);
            osc.stop(now + 0.3);
        } else {
            clearInterval(modemInterval);
        }
    }, 2000);
}
```

Action de déclenchement du son lors du clic et initialisation du moteur audio.

```
soundToggle.addEventListener('click', () => {
    soundEnabled = !soundEnabled;

    if (soundEnabled) {
        try {
            if (!audioContext) {
                audioContext = new (window.AudioContext ||
window.webkitAudioContext)();
            }

            // Test immédiat d'un son
            const now = audioContext.currentTime;
            const osc = audioContext.createOscillator();
            const gain = audioContext.createGain();

            osc.connect(gain);
            gain.connect(audioContext.destination);

            osc.frequency.value = 1500;
            osc.type = 'sine';
            gain.gain.setValueAtTime(0.3, now);
            gain.gain.exponentialRampToValueAtTime(0.01, now + 0.2);
```

```
            osc.start(now);
            osc.stop(now + 0.2);
```

Changement du bouton son.

```
startSounds();
            soundToggle.textContent = '🔇 DÉSACTIVER SON';
            soundToggle.style.background = '#00ff00';
            soundToggle.style.color = '#ff0000';
        } catch(e) {
            console.error('Erreur audio:', e);
        }
    } else {
        soundEnabled = false;
        soundToggle.textContent = '🔊 ACTIVER SON';
        soundToggle.style.background = '#ff0000';
        soundToggle.style.color = '#000';
    }
});
```

Fonction du compteur qui s’incrémente automatiquement.

```
function siphonEuros() {
    // Simule le pompage de tokens Euros (entre 10 000 et 99 999 tokens par
intervalle)
    let tokensToSiphon = Math.random() * 89999 + 10000;

    montantEuros += tokensToSiphon;

    // Affichage avec 8 décimales et formatage des milliers pour plus de
lisibilité
    counter.textContent = "FONDS EXTORQUÉS : " +
montantEuros.toLocaleString('en-US', {minimumFractionDigits: 8,
maximumFractionDigits: 8}) + " EUROS";
}

// Mise à jour rapide (toutes les 200 ms) pour un flux dynamique
setInterval(siphonEuros, 200);
```

Si vous remarquez des erreurs, sachez qu'elles sont intentionnelles !

Merci !